TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing: *How did we get here?*

Wes J. Lloyd
School of Engineering and Technology
University of Washington - Tacoma

1

FEEDBACK FROM 9/30

- Perspective on material: 6.8 (→ *mostly new to me*)
- Pace: 5.3 (→ *slightly fast*)
- 24 respondents
- SIMD Vector arithmetic example
  - Comment: *"SIMD faster than MIMD in some cases, I can understand, but cannot logically explain it"*
- SIMS vs. MIMD
- SIMD instruction relationships with CPU development
  - First introduced with the Intel Pentium III in 1999

2

FEEDBACK - 2

- Flynn's machine (Flynn's taxonomy):
  - Ideal is to classify HW architectures as either: SISD, SIMD, or MIMD
    - Example question could describe a computer or computing scenario as ask if it is SISD, SIMD, or MIMD…
- What are some advantages of using heterogeneous HW vs. homogeneous HW?
- Roofline explanation

3

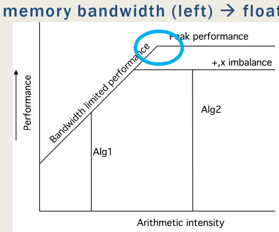
ARITHMETIC INTENSITY

- **Arithmetic intensity**: Ratio of work (W) to memory traffic r/w (Q) $I = \frac{W}{Q}$
  Example: # of floating point ops per byte of data read
- Characterizes application scalability with SIMD support
  - *SIMD can perform many fast matrix operations in parallel*
- ***High arithmetic Intensity:***
  Programs with dense matrix operations scale up nicely (many calcs vs memory RW, supports lots of parallelism)
- ***Low arithmetic intensity:***
  Programs with sparse matrix operations do not scale well with problem size
  (memory RW becomes bottleneck, not enough ops!)

4

ROOFLINE MODEL

- When program reaches a given arithmetic intensity performance of code running on CPU hits a "roof"
- CPU performance bottleneck changes from: memory bandwidth (left) → floating point performance (right)



Key take-aways:
When a program's has **low** Arithmetic Intensity, memory bandwidth limits performance..

With **high** Arithmetic intensity, the system has peak parallel performance…
→ *performance is limited by??*

5

FEEDBACK - 3

- Is a GPU considered a super CPU based on the structure?
  - *Differences between CPUs and GPUs*
  - CPU: composed of just a few cores with lots of cache memory that can handle a few software threads at a time.
  - GPU: composed of hundreds of cores that can handle thousands of threads simultaneously. Ability of a GPU with 100+ cores to process thousands of threads can accelerate ***some software*** by 100x over a CPU alone. GPU achieves acceleration while being more power- and cost-efficient than a CPU.
  - *CPU: emphasis is low latency*
  - *GPU: emphasis is high throughput*

6

FEEDBACK - 3

- Are class activities included in grading?
- Panopto recordings

7

OBJECTIVES

- **Cloud Computing: How did we get here?**
  - *Parallel and distributed systems (Marinescu Ch. 2: 1st edition, or Ch. 4: 2nd edition)*
  - Data, thread-level, task-level parallelism
  - Parallel architectures
  - SIMD architectures, vector processing, multimedia extensions
  - Graphics processing units
  - Speed-up, Amdahl's Law, Scaled Speedup
  - Properties of distributed systems
  - Modularity
  - Functional vs. Non-functional requirements

8

DISTRIBUTED SYSTEMS

- Collection of autonomous computers, connected through a network with distribution software called "middleware" that enables coordination of activities and sharing of resources
- **Key characteristics:**
- Users perceive system as a single, integrated computing facility.
- Compute nodes are autonomous
- Scheduling, resource management, and security implemented by every node
- Multiple points of control and failure
- Nodes may not be accessible at all times
- System can be scaled by adding additional nodes
- Availability is realized at different levels: HW, software, network providing different reliability

9

DISTRIBUTED SYSTEMS - 2

- Key non-functional attributes
  - Known as "ilities" in software engineering
- Availability – 24/7 access?
- Reliability - Fault tolerance
- Accessibility – reachable?
- Usability – user friendly
- Understandability – can under
- Scalability – responds to variable demand
- Extensibility – can be easily modified, extended
- Maintainability – can be easily fixed
- Consistency – data is replicated correctly in timely manner

10

TRANSPARENCY PROPERTIES OF DISTRIBUTED SYSTEMS

- **Access transparency**: local and remote objects accessed using identical operations
- **Location transparency**: objects accessed w/o knowledge of their location.
- **Concurrency transparency**: several processes run concurrently using shared objects w/o interference among them
- **Replication transparency**: multiple instances of objects are used to increase reliability
  - users are unaware if and how the system is replicated
- **Failure transparency**: concealment of faults
- **Migration transparency**: objects are moved w/o affecting operations performed on them
- **Performance transparency**: system can be reconfigured based on load and quality of service requirements
- **Scaling transparency**: system and applications can scale w/o change in system structure and w/o affecting applications

11

TYPES OF MODULARITY

- ***Soft modularity:*** TRADITIONAL
- Divide a program into modules (classes) that call each other and communicate with shared-memory
- A procedure calling convention is used (or method invocation)
- Object-oriented programming classic best practices:
- **Minimize coupling** between classes (OO) and modules
- **Maximize cohesion** between functions in classes (OO) and modules
  - Best practices lead to improved software reusability, maintainability, portability

12

TYPES OF MODULARITY - 2

- ***Enforced modularity:*** CLOUD COMPUTING
- Program is divided into modules that communicate only through message passing
- The ubiquitous ***client-server*** paradigm
- Clients and servers are independent decoupled modules
- System is more robust if servers are stateless
- May be scaled and deployed separately
- May also FAIL separately!

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.13

13

COUPLING AND COHESION

- **Object-oriented coupling**
- Degree of interdependence between software modules
- A measure of how connected two classes or modules are
- Captures the degree of the relationships between modules
- Coupling is usually contrasted with cohesion
- Low coupling often correlates with high cohesion
- High coupling often correlates with low cohesion

- **Object-oriented cohesion**
- Degree to which elements inside a class or module belong together
- Do the methods and data inside of a class interoperate with each other (***High cohesion***)? Or is the class a catch all bin of random functions (***Low cohesion***)?
  - E.g. "Util" class where random helper routines land… (***low cohesion***)

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.14

14

FUNCTIONAL VS. NON-FUNCTIONAL ATTRIBUTES OF SYSTEMS

- **Functional requirement:**
- Pertains to a system supporting a specific function
- What a system is supposed to do
- Testable with unit tests, integration tests, etc.

- **Non-functional requirement:**
- Specifies criteria used to judge how a system operates
- How a system should be (or behave)
- Considered as "quality" attributes of systems
- Testable by applying metrics to characterize degree of possessing a given quality

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.15

15

NON-FUNCTIONAL REQUIREMENT: HIGH AVAILABILITY

- ***The system should be highly available.***
- The system should be **99.9%** available per month
  - Maximum downtime: **43m 49.7s** monthly, **8hr 45min 36s** yearly
  - Functional attribute: system should notify users if there is an issue affects the availability or may cause downtime.
- Availability equation:

$$\text{AVAILABILITY} = \frac{\text{MTBF}}{\text{MTBF}+\text{MTTR}}$$

- MTBF: Mean time between failures
- MTTR: Mean time to Repair
- MTBF = ~1 month = 43,757 min; MTTR = 43 min
- AVAILABILITY = 43757 / 43800 = 99.9018265%

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.16

16

STRATEGIES FOR HIGH AVAILABILITY

- Replicate system resources in multiple data centers or cloud computing regions
- Use redundant infrastructure components
  - For load balancing, fault tolerance
- Report availability status via portal
- Allow users to immediately report outages
- Notification systems to alert system admins when system experiences an outage

- **Tradeoffs:**
- Highly available cloud resources are more expensive
- Replicating app components (e.g. database) adds cost

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.17

17

QUANTIFYING NON-FUNCTIONAL QUALITY ATTRIBUTES

- What are the "best" metrics to quantify non-functional quality attributes?
  - Consider ease/effort/time/cost of assessment
  - Relationship to expert opinion (e.g. correlation)
  - Relationship to other measures

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.18

18

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS

- Multi-core CPU technology and hyper-threading
- What is a
  - Heterogeneous system?
  - Homogeneous system?
  - Autonomous or self-organizing system?
- **Fine grained vs. coarse grained parallelism**
- Parallel message passing code is easier to debug than shared memory (e.g. p-threads)
- Know your application's max/avg **Thread Level Parallelism** (***TLP***)
- **Data-level parallelism**: Map-Reduce, (SIMD) Single Instruction Multiple Data, Vector processing & GPUs

19

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 2

- **Bit-level parallelism**
- **Instruction-level parallelism** (CPU pipelining)
- **Flynn's taxonomy**: computer system architecture classification
  - **SISD** – Single Instruction, Single Data (modern core of a CPU)
  - **SIMD** – Single Instruction, Multiple Data (Data parallelism)
  - **MIMD** – Multiple Instruction, Multiple Data
  - MISD is RARE; application for fault tolerance…
- **Arithmetic intensity**: ratio of calculations vs memory RW
- **Roofline model:** Memory bottleneck with low arithmetic intensity
- **GPUs**: ideal for programs with high arithmetic intensity
  - SIMD and Vector processing supported by many large registers

20

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 3

- **Speed-up (S)**
  $S(N) = T(1) / T(N)$
- **Amdahl's law:**
  $S = 1/ \alpha$
  α = percent of program that must be sequential
- **Scaled speedup with N processes:**
  $S(N) = N – \alpha(N-1)$
- Moore's Law
- Symmetric core, Asymmetric core, Dynamic core CPU
- Distributed Systems Non-function quality attributes
- Distributed Systems – Types of Transparency
- Types of modularity- Soft, Enforced
- Functional vs. Non-functional requirements

21

INTRODUCTION TO CLOUD COMPUTING


22

OBJECTIVES - 2

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

23

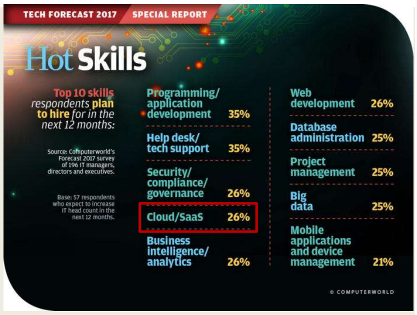
WHY STUDY CLOUD COMPUTING?

- LINKEDIN - TOP IT Skills from job app data
  - #1 Cloud and Distributed Computing
  - https://learning.linkedin.com/week-of-learning/top-skills
  - #2 Statistical Analysis and Data Mining
- FORBES Survey – 6 Tech Skills That'll Help You Earn More
  - #1 Data Science
  - #2 Cloud and Distributed Computing
  - http://www.forbes.com/sites/laurencebradford/2016/12/19/6-tech-skills-thatll-help-you-earn-more-in-2017/

24

WHY STUDY CLOUD COMPUTING? - 2

- Computerworld Magazine



October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.25

25

A BRIEF HISTORY OF CLOUD COMPUTING

- John McCarthy, 1961
  - Turing award winner for contributions to AI
- "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility… The computer utility could become the basis of a new and important industry…"

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.26

26

CLOUD HISTORY - 2

- Internet based computer utilities
- Since the mid-1990s
- Search engines: Yahoo!, Google, Bing
- Email: Hotmail, Gmail

- 2000s
- Social networking platforms: MySpace, Facebook, LinkedIn
- Social media: Twitter, YouTube

- Popularized core concepts
- Formed basis of cloud computing

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.27

27

CLOUD HISTORY: SERVICES - 1

- Late 1990s – Early Software-as-a-Service (SaaS)
  - Salesforce: Remotely provisioned services for the enterprise
- 2002 -
  - Amazon Web Services (AWS) platform: Enterprise oriented services for remotely provisioned storage, computing resources, and business functionality
- 2006 – **Infrastructure-as-a-Service (IaaS)**
  - Amazon launches Elastic Compute Cloud (EC2) service
  - Organization can "lease" computing capacity and processing power to host enterprise applications
  - Infrastructure

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.28

28

CLOUD HISTORY: SERVICES - 2

- 2006 – **Software-as-a-Service (SaaS)**
  - Google: Offers Google DOCS, "MS Office" like fully-web based application for online documentation creation and collaboration
- 2009 – **Platform-as-a-Service (PaaS)**
  - Google: Offers Google App Engine, publicly hosted platform for hosting scalable web applications on google-hosted datacenters

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.29

29

CLOUD COMPUTING NIST GENERAL DEFINITION

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and reused with minimal management effort or service provider interaction"…

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.30

30

MORE CONCISE DEFINITION

"Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources."

From Cloud Computing Concepts, Technology, and Architecture
Z. Mahmood, R. Puttini, Prentice Hall, 5th printing, 2015

October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.31

31

BUSINESS DRIVERS FOR CLOUD COMPUTING

- Capacity planning
- Cost reduction
- Operational overhead
- Organizational agility

October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.32

32

BUSINESS DRIVERS FOR CLOUD COMPUTING

- Capacity planning
  - Process of determining and fulfilling future demand for IT resources
  - Capacity vs. demand
  - Discrepancy between capacity of IT resources and actual demand
  - Over-provisioning: resource capacity exceeds demand
  - Under-provisioning: demand exceeds resource capacity
  - Capacity planning aims to minimize the discrepancy of available resources vs. demand

October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.33

33

Dwight, The Office TV sitcom

October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.34

34

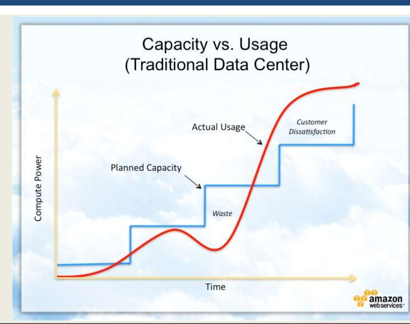
BUSINESS DRIVERS FOR CLOUD - 2

- Capacity planning
  - Over-provisioning: is costly due to too much infrastructure
  - Under-provisioning: is costly due to potential for business loss from poor quality of service
- Capacity planning strategies
  - Lead strategy: add capacity in anticipation of demand (pre-provisioning)
  - Lag strategy: add capacity when capacity is fully leveraged
  - Match strategy: add capacity in small increments as demand increases
- Load prediction
  - Capacity planning helps anticipate demand flucations

October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.35

35

CAPACITY PLANNING

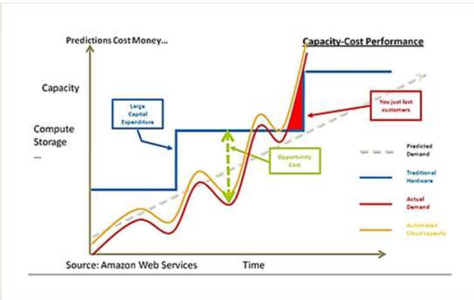


October 2, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L3.36

36

CAPACITY PLANNING - 2

- Ca



October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.37

37

BUSINESS DRIVERS FOR CLOUD - 3

- Cost reduction
  - IT Infrastructure acquisition
  - IT Infrastructure maintenance
- Operational overhead
  - Technical personnel to maintain physical IT infrastructure
  - System upgrades, patches that add testing to deployment cycles
  - Utility bills, capital investments for power and cooling
  - Security and access control measures for server rooms
  - Admin and accounting staff to track licenses, support agreements, purchases

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.38

38

BUSINESS DRIVERS FOR CLOUD - 4

- Organizational agility
  - Ability to adapt and evolve infrastructure to face change from internal and external business factors
  - Funding constraints can lead to insufficient on premise IT
  - Cloud computing enables IT resources to scale with a lower financial commitment

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.39

39

TECHNOLOGY INNOVATIONS LEADING TO CLOUD

- Cluster computing
- Grid computing
- Virtualization
- Others

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.40

40

CLUSTER COMPUTING

- Cluster computing (clustering)
  - Cluster is a group of independent IT resources interconnected as a single system
  - Servers configured with homogeneous hardware and software
    - Identical or similar RAM, CPU, HDDs
  - Design emphasizes redundancy as server components are easily interchanged to keep overall system running
    - Example: if a RAID card fails on a key server, the card can be swapped from another redundant server
  - Enables warm replica servers
    - Duplication of key infrastructure servers to provide HW failover to ensure high availability (HA)

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.41

41

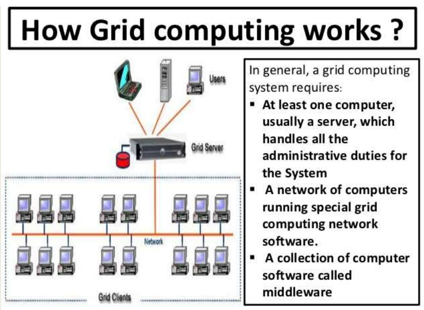
GRID COMPUTING

- On going research area since early 1990s
- Distributed heterogeneous computing resources organized into logical pools of loosely coupled resources
- For example: heterogeneous servers connected by the internet
- Resources are heterogeneous and geographically dispersed
- Grids use middleware software layer to support workload distribution and coordination functions
- Aspects: load balancing, failover control, autonomic configuration management
- Grids have influenced clouds contributing common features: networked access to machines, resource pooling, scalability, and resiliency

October 2, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.42

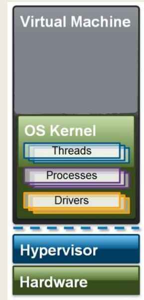
42

GRID COMPUTING - 2



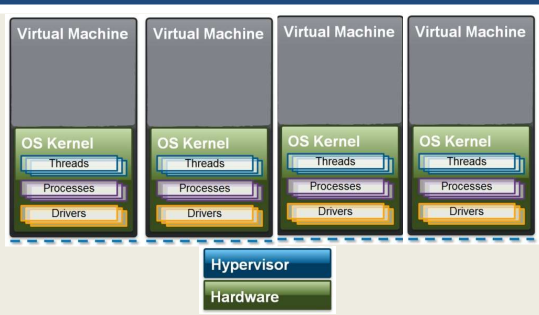
43

VIRTUALIZATION



44

VIRTUALIZATION



45

VIRTUALIZATION

- Simulate physical hardware resources via software
  - The virtual machine (virtual computer)
  - Virtual local area network (VLAN)
  - Virtual hard disk
  - Virtual network attached storage array (NAS)
- Early incarnations featured significant performance, reliability, and scalability challenges
- CPU and other HW enhancements have minimized performance GAPs

46

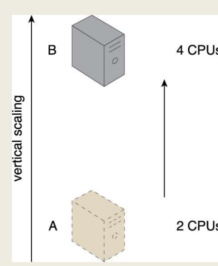
KEY TERMINOLOGY

- **On-Premise Infrastructure**
  - Local server infrastructure not configured as a cloud
- **Cloud Provider**
  - Corporation or private organization responsible for maintaining cloud
- **Cloud Consumer**
  - User of cloud services
- **Scaling**
  - **Vertical scaling**
    - Scale up: increase resources of a single virtual server
    - Scale down: decrease resources of a single virtual server
  - **Horizontal scaling**
    - Scale out: increase number of virtual servers
    - Scale in: decrease number of virtual servers

47

VERTICAL SCALING

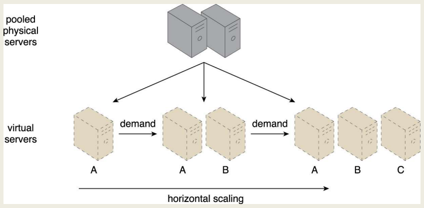
- Reconfigure virtual machine to have different resources:
  - CPU cores
  - RAM
  - HDD/SDD capacity
- May require VM migration if physical host machine resources are exceeded



48

HORIZONTAL SCALING

- Increase (scale-out) or decrease (scale-in) number of virtual servers based on demand



49

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers

50

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available

51

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed

52

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional servers required

53

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional servers required
Not limited by individual server capacity	Limited by individual server capacity

54

KEY TERMINOLOGY - 2

- Cloud services
  - Broad array of resources accessible “as-a-service”
  - Categorized as Infrastructure (IaaS), Platform (PaaS), Software (SaaS)
- Service-level-agreements (SLAs):
  - Establish expectations for: uptime, security, availability, reliability, and performance

55

GOALS AND BENEFITS

- **Cloud providers**
  - Leverage economies of scale through mass-acquisition and management of large-scale IT resources
  - Locate datacenters to optimize costs where electricity is low
- **Cloud consumers**
  - Key business/accounting difference:
  - **Cloud computing enables anticipated capital expenditures to be replaced with operational expenditures**
  - Operational expenditures always scale with the business
  - Eliminates need to invest in server infrastructure based on anticipated business needs
  - Businesses become more agile and lower their financial risks by eliminating large capital investments in physical infrastructure

56

CLOUD BENEFITS - 2

- On demand access to pay-as-you-go resources on a short-term basis (less commitment)
- Ability to acquire “unlimited” computing resources on demand when required for business needs
- Ability to add/remove IT resources at a fine-grained level
- Abstraction of server infrastructure so applications deployments are not dependent on specific locations, hardware, etc.
  - The cloud has made our software deployments more agile...

57

CLOUD BENEFITS - 3

- Example: Using 100 servers for 1 hour costs the same as using 1 server for 100 hours
- Rosetta Protein Folding: Working with a UW-Tacoma graduate student, we recently deployed this science model across 5,900 compute cores on Amazon for 2-days...
- ***What is the cost to purchase 5,900 compute cores?***
- Recent Dell Server purchase example: 20 cores on 2 servers for $4,478...
- Using this ratio 5,900 cores costs $1.3 million (purchase only)

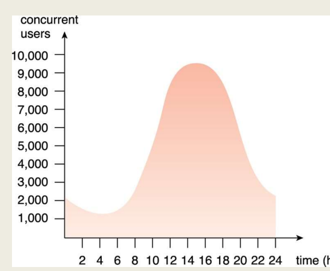
58

Gene Wilder, Charlie and the Chocolate Factory

59

CLOUD BENEFITS

- Increased scalability
  - Example demand over a 24-hour day →
- Increased availability
- Increased reliability



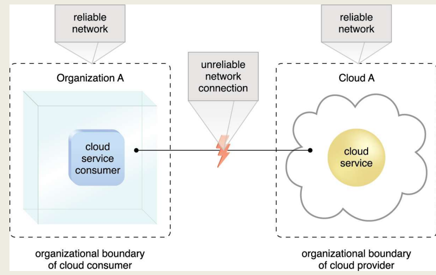
60

CLOUD ADOPTION RISKS

- **Increased security vulnerabilities**
  - Expansion of trust boundaries now include the external cloud
  - Security responsibility shared with cloud provider
- **Reduced operational governance / control**
  - Users have less control of physical hardware
  - Cloud user does not directly control resources to ensure quality-of-service
  - Infrastructure management is abstracted
  - Quality and stability of resources can vary
  - Network latency costs and variability

61

NETWORK LATENCY COSTS



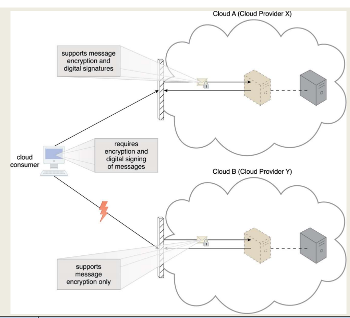
62

CLOUD RISKS - 2

- **Performance monitoring of cloud applications**
  - Cloud metrics (AWS cloudwatch) support monitoring cloud infrastructure (network load, CPU utilization, I/O)
  - Performance of cloud applications depends on the health of aggregated cloud resources working together
  - User must monitor this aggregate performance
- **Limited portability among clouds**
  - Early cloud systems have significant “vendor” lock-in
  - Common APIs and deployment models are slow to evolve
  - Operating system containers help make applications more portable, but containers still must be deployed
- **Geographical issues**
  - Abstraction of cloud location leads to legal challenges with respect to laws for data privacy and storage

63

CLOUD: VENDOR LOCK-IN



64

QUESTIONS

65